Recovery from Bad Files

Remedy procedure

Friday, September 7, 2001

From time to time, a node exhibits unknown allocated areas in nonvolatile memory as determined by the page application PAGENVOL. This application scans through the nonvolatile memory structures matching each area with a corresponding entry in the CODES table. An unknown block is one that has no reference in the CODES table, which is a kind of file directory for the nonvolatile memory file system. This note describes a procedure for recovery from such unknown blocks.

When PAGENVOL detects an unknown area, it shows some relevant data in the lower area of the screen that describes where it was found. There are three long words shown. The first is the address in nonvolatile memory of the unknown block. Any allocated area in nonvolatile memory begins with an 8-byte header that precedes the contents of the file. This header consists of 2 long words, which are displayed following the address of the block. The two long words are the size of the block (including the 8-byte header) and the type of the block, which is normally 0000000F.

Here is an unknown block example from recent experience:

```
0046B7B0  000003C8  0000000F
```

This data indicates that an unknown block as found at 0046B7B0, where one can find the two long words 000003C8 and 0000000F.

The remedy for recovering from this situation involves manually creating an entry in the CODES table that refers to the unknown block, then using the Download page application to free it.

Find an unused entry in the CODES table, which for IRMs is normally located at 00407000—004077FF. Each entry in CODES is 32 bytes, with the following layout, shown as 8-bytes per line:

Generic pattern		*Example*	
typeName	fileName	50414745	4D444D50
fileSize	checksum	000026F6	0612AD08
fileAddr	execAddr	00460730	00000000
file-version-date		01010211	33200421

In an unused entry in the CODES table, one that is all zeros, manually install an entry that includes a typeName of AAAA, which is 41414141, a fileSize that *is 8 bytes less* than the displayed size long word, and a data address that is *8 bytes more* than the displayed block address. For the case of the unknown block example above, the entry to be installed would be as follows:

```
    41414141  00000000
    000003C0  00000000
    0046B7B8  00000000
    00000000  00000000
```

Now go to Page `D`, the download page and get a `DIR` listing of all the files whose `typeName` begins with `A`. There should only be one file listed, since there is no `typeName` in use that begins with letter `A`. In this case, the list should appear as:

```
    AAAA@@@@ 03C0 D   00/00/00 0000
```

Free the (manually) allocated block by typing `0000` over the `03C0` size field and clicking with the cursor immediately after the last `0` of that field. The Page `D` display is immediately updated with blanks to show that the entry freed no longer exists. If you look in memory at the manually entered `CODES` table entry, it should now be all zeros.

Running the `PAGENVOL` application, there should now be one less unknown block found. If a second unknown block had existed before this remedy, it should now appear on the screen. The data for only a single unknown block is shown at one time, although there is also a count of such blocks shown after `UNKN=`. Repeat the remedy procedure for each unknown block shown.

One may hope that one day this remedy might be automated. There is no known reason why unknown blocks should be formed during ordinary system operation.